On Effect Of Faults In Vibration Control Of Fairing Structures

Young Man Kim* , Anish Arora*, Vinod Krishnan K*, Sandeep Kulkarni† and Umamaheshwaran A†
* Department of Computer and Information Science
The Ohio State University
Columbus, OH 43210, USA
Email:{kimyoun, anish, vinodkri}@cis.ohio-state.edu
† Department of Computer Science
Michigan State University

Abstract

efects in various components or within the controller of an automated system, can be amplified by a closed-loop system and ultimately bring down the plant itself. Hence it is important to incorporate fault-tolerance into such systems. Before designing a fault-tolerant control system, it is important to understand the different kinds of faults that the system is vulnerable to and the effect they can have on the system. In this paper, we consider the Boeing Open Experimental Platform fairing control application to study the effect of faults. We also present alternate control schemes and compare the effect of faults under those schemes. efects in various components or within the controller of an automated system, can be amplified by a closed-loop system and ultimately bring down the plant itself. Hence it is important to incorporate fault-tolerance into such systems. Before designing a fault-tolerant control system, it is important to understand the different kinds of faults that the system is vulnerable to and the effect they can have on the system. In this paper, we consider the Boeing Open Experimental Platform fairing control application to study the effect of faults. We also present alternate control schemes and compare the effect of faults under those schemes. D

This work was partially sponsored by DARPA contract OSU-RF #F33615-01-C-1901, NSF grant NSF-CCR-9972368, an Ameritech Faculty Fellowship, and two grants from Microsoft Research.

On leave from the Department of Computer Science, Kookmin University, Seoul, South Korea

I. Introduction

Automated systems are vulnerable to faults. Defects in various components or within the controller, can be amplified by a closed-loop system and ultimately bring down the plant itself. A cost effective way to obtain increased dependability in automated systems is to introduce fault-tolerant control [1]. The goal is to prevent local faults from developing into failures that can stop production or cause safety hazards.

Before designing a fault-tolerant control system, it is important to understand the different kinds of faults that the system is vulnerable to and the effect they can have on the system. By way of an example of a control application, we consider the control of acoustic vibrations in fairing structures. The objective of the application is to reduce the vibroacoustic environment inside launch vehicle payload fairings during liftoff and ascent to orbit. The application of active control to vibration structures has been studied from many years. The earlier systems were mostly centralized in control. However, the extension of this technology to large scale systems motivates the design of control that is distributed. The vehicle subsystem that we consider is characterized by 100s to 10000s of nodes driving sensors and actuators trying to acheive fine grain control.

Faults in such a system can occur at different levels in the system. At the platform level sensors, actuators and other components can stop functioning or they can exhibit random behavior. The middleware level that provides underlying services such as controller group synchronization, communication, reconfiguration, fault-detection etc., could introduce vulnerabilities such as unpredictable delays or incorrect detection. In this paper, we present the effect of various types of faults on the performance of the distributed fairing control application. This study throws light on answers to important research questions such as: At what level should fault-tolerance be added to the control? Several recent works deal with faults in control systems via detection and subsequent isolation. Is such a detection based tolerance feasible? (Especially since the detection services themselves are vulnerable to faults). How to design fault-tolerance in control? Does a purely local scheme provide better fault-tolerance? How does a linear control scheme compare to an on-off or bang bang control scheme? How many sensor nodes are needed for the control?

Specifically in this paper, we consider the Boeing Open Experimental Platform fairing control application to study the effect of faults. The architecture of this platform is described in the following section. We also present alternate control schemes and compare the effect of faults under those schemes.

Organization of the paper In section 2, we describe the architecture of the Boeing Open Experimental Platform fairimg control application. In section 3, we describe the experimental setup and define metrics that we use for performance evaluation. In section 4, we enlist different types of faults and study the effect of those on the Boeing control application. In section 5, we present alternate fault-tolerant control schemes and compare those with the Boeing control scheme. In section 6, we present conclusions and goals for future research.

II. Boeing Platform Simulator

This section describes Boeing's Open Experimental Platform [2] for an acoustic and structural damping application. The objective of this application is to reduce acoustic vibrations inside fairing shaped space launch vehicles during their ascent to orbit.

A. Hardware Testbed Description

The experimental testbed in the Boeing OEP is a scaled down version (about 1/16th) of a typical fairing shaped space launch vehicle (hereby referred to as a fairing). Six speakers installed around the fairing approximate the field vibrations. A network of PVDF sensors are attached to the fairing to map the response of the fairing to acoustic inputs. Piezo-electric actuators control the structural response of the fairing.

B. The Simulated Environment

The simulated environment of the Boeing OEP application includes the following:

1. A simulation of the fairing structure, using 100 computational nodes. Each node comprises a sensor, an actuator and a processing element.
2. A hierarchical control capability to adapt the system responses to changes in vibration modes of the fairing.

3. Processes for detecting and reacting to node failures.

The 100 node system is partitioned into several groups. Each group acts to damp a particular mode of vibration in the fairing. The number of nodes assigned to each mode depends on the frequency of that mode and the energy required to acheive the dampening. There can be as many as 20 modes to address. The system functionality is divided into 4 main categories: low level control, group control, system identification and configuration. These are described below.

1. Low Level Control

Low level control performs actions associated with an individual node. It detects the dispacement of the fairing and damps vibration by providing a command output to the actuator. The natural frequencies of the targeted modes are usually of the order of 200 Hz. To mitigate the effect of sampling and computation related delays on the damping process, the controller works at 2Khz. The control algorithm first filters the sensor data to remove the data outside the frequency band of interest. The compensator process is an output feedback control scheme. The output is fed to an actuator. The actuator command processing is based on 2 inputs. One input is from the low level controller. The other input is the ping profile defined by the system identiifcation function. The controller actuation command is combined with the ping command to generate the total actuator command. Each node also provides health status information to the group control function at a frequency of 100Hz. The node estimates the effectiveness of damping by comparing it with an expected behavior, i.e., a health model.

2. Group Control

The group control function gathers health information from all individual nodes and provides it to the configuration function. The group control function also collects all actuator and sensor data from individual nodes and sends it to the system identification function. Updated modal characteristic data is received from the system identification function. This data is refined as appropriate for use by low level control algorithms and the modified control parameters are sent to individual nodes. Individual group members are also activated or deactivated based on commands from the configuration function.

3. System Identification Function

The significant modal frequencies of the fairing keeps changing as a function of the flight conditions. The system identification function is responsible for determining the current fairing vibroacoustic modal characteristics. This is acheived by a set of commands to ping selected actuators and then measuring the response from corresponding sensors. The pinging is performed based on information received from group controller nodes regarding data from each member. In the Boeing OEP, pinging is performed every 1 second. Since the sampling rate is 2Khz, a total of 4000 values from sensors and actuators are stored over this interval. The data obtained from pinging is used to compute modal characteristics such as shapes, frequencies and damping parameters associated with each vibroacoustic mode of interest. This information is then fed to the configuration function.

4. Configuration function

Based on information obtained from the group control function regarding node health status and the vibroacoustic modal characteristic data obtained from the system identification function, the configuration module defines nodes that are more effective in controlling each mode of interest. Thus it re-optimizes group membership and also modifies group configuration and/or low level parameters. This information is then provided to the group control function.

C. Timing Of The System-Id Loop

The ping frequency is 1Hz. Transfer of data to the system-id function takes one second. The system-id and configuration calculations take 2.5 seconds. The movement of modified parameters back to low level controllers takes 0.5 seconds. Thus, the system identification loop runs at a frequency of 0.2 Hz. Once every 5 seconds a ping action is performed and group membership and low level parameters are re-optimized.

III. Experimental Setup And Metrics

In this section, we describe the configuration of the Boeing OEP platform under which all our experiments were performed. We also explain the performance metrics that we use for our analysis.

A. Experimental Setup

In the fairing simulator there are 100 physical nodes that are capable of low level actuation. In the configuration that we use, there is 1 configurator node, 1 system id node, 11 group controller nodes, and 87 low level controller nodes. (A mapping of nodes to their physical locations is provided in the appendix.) At the beginning of each experiment, an optimum node-mode assignment is assumed to be made.

B. Performance Metrics

The sum of the squares of structural velocities at the 100 nodes is taken as a measure of the total energy in the fairing. Let EU be the energy at a given instant in the fairing without any control being applied. Let EC be the energy in the fairing at a given instant with control being applied. The absolute metric for a given experiment is the energy reduction ratio during the experiment.
Energy Reduction Ratio (ERR) = Mean EC / Mean EU
In order to compare results from different fault experiments, we define the following metrics:
Performnace Degradation Ratio (R) for experiment X = R1 / R2
R1 = Mean EC / Mean EU during fault experiment X
R2 = Mean EC / Mean EU during experiment without any fault

C. Qualitative Performance Index

In the configuration that we operate the Boeing OEP, R2 is found to be 0.17 on an average over 50 experiments without any fault. Based on this measure, we define the following qualitative performance indices for any given fault experiment:
$R1 < 0.45 = Tolerable$
$0.45 < R1 < 0.75 = Significantdegradation$
$0.75 < R1 < 1.25 = Substantialdegradation$
$R1 > 1.25 = Intolerable$

IV. Fault Types Studied

In this section, we describe the different types of faults whose effects we study on the Boeing OEP. We classify these faults broadly into 2 categories: platform level and network level.

A. Platform Level Faults

Platform level faults are those that occur on the hardware or software at the low level. The improper functioning of components at the low level could manifest itself at the high level in vastly different ways. Some examples of these are described below:

1. Fail-stop / Crash Faults

Sensors, actuators or the processing element at a node simply stop functioning. Reasons could be power failure at the node, physical damage etc. Communication of this node with others may be cut-off. These faults are sometimes detectable (fail-stop fault) and sometimes not (crash fault). In our experiments, we study the effect of different nodes crashing on the Boeing OEP. Crashed nodes are assumed incapable of communicating with others. Hence these nodes are not detected as dead for the higher layers to take corrective action.

2. Random and Stuck-at Faults

Certain hardware problems or a fault at the hardware-software interface can cause sensors and actuators to behave in an arbitrary manner. Random sensors return an arbitray reading. Random actuators perform an arbitrary action.

We specify the randomness of a node by providing a mean and standard deviation for the values. A special case of the random fault category is a stuck-st fault. Stuck-at sensors return a constant specified value. Stuck-at actuators always apply a constant specified force.

3. (Intelligent) Byzantine Faults

In order to study the effect of low level faults that can cause maximum harm to the system, we program certain nodes to behave to generate worst possible signal values. These Byzantine nodes generate values of maximum magnitude in a direction opposite to the correct one. Such Byzantine faults could also arise due to security loop holes in the system.

4. Debonding Faults

Under extensive vibrations, sensors and actuators can physically seperate from their contacts on the fairing. So sensors and actuators are not fully effective. They perform only at a fraction of their capability. To study the effect of debonding faults, we specify debonding of a node in terms of the final effectiveness percentage and the rate of debonding.

B. Network Level Faults

The network that handles communication between different nodes is itself subject to faults. We model these faults using delay in communication. For example, when system-id signals reconfiguration, different nodes could be getting these parameters at different times. We study the effect of such delays and jitter on the quality of control obtained. We added the ability to introduce message delays in the Boeing OEP by re-implementing certain functions in the OEP using threads to simulate delays.

V. Fault Experiments

In this section, we describe our experiments to study the effect of faults listed in section 4, and summarize the results. As described before, in the fairing simulator there are 100 physical nodes that are capable of low level actuation. In the configuration that we use, there is 1 configurator node, 1 system id node, 11 group controller nodes, and 87 low level controller nodes. (A mapping of nodes to their physical locations is provided in the appendix.) At the beginning of each experiment, an optimum node-mode assignment is assumed to be made. This optimum assignment is also shown in the appendix. Note that modes 2 to 9 have 10 nodes assigned to them. Mode 1 has 3 nodes, and modes 10 and 11 have 2 nodes each assigned to them. These assignments can change at every system-id cycle. Initially the low level controllers are not active. They get activated upon receiving control parameters from the group controller. The faults are specified to the system by means of an input file. This file contains the ids of the faulty nodes, the type of the fault and the duration of the fault. Note that each experiment was performed 10 times and the resulting metrics were averaged. These results are described below.

A. Crash Faults

Here, we study the effect of nodes that simply stop functioning. Communication of these nodes is cut off from the rest of the system. Hence these failures are not detected. The case of the low level nodes crashing is distinguished from that of the group controllers crashing since their effects could be different.

a) Observation 1: When 10 low level controllers from different modes crash, the result is tolerable degradation. When all the 10 controllers belong to the same mode, thus one mode left completely uncontrolled, the result is significant degradation.

The 10 low level controller nodes from different modes were chosen randomly. Each mode had atleast one node active. For testing the effect of 10 crashed controllers that belong to the same mode, the experiment was repeated for each mode seperately. The worst degradation results amongst those are presented. From the observations it is clear that having atleast one node active per mode yields tolerable control quality.

b) Observation 2: The crashing of 1 or 2 group controllers results in significant degradation. Crashing of 3 group controllers results in substantial degradation.

Group controller nodes are responsible supplying the configurator with node health values and also supplying low level nodes with modified control parameters in every system-id cycle. The system loses this functionality upon the group controllers crashing. Thus when a group controller crashes, all the low level nodes continue with the old and now incorrect parameters. If the group controller crashes before the low level nodes are activated, its low level controllers offer no control. Experiments were repeated with different group controllers crashing and at different times in a system-id cycle.

c) Recommendation: Detecting group controller crashes is important. Rotating group controllers after every system-id cycle will make the impact short-term.

B. Random And Stuck-at Faults

Based on our experiments without any fault, the operating range of sensors and actuators is reported in the table x, in the form of maximum and average values. In this section we first describe the effect of sensors and actuators generating random values that are close to their operating range. Detecting a bad sensor or actuator is extremely difficult in this case. We consider sensors and actuators belonging to same and different modes. We then describe the effect of sensors and actuators producing random values that can be anywhere in the range that they are capable of generating.

d) Observation 3: When sensors and actuators generate random values entirely within the range of operation, the degradation of control is tolerable. However when the range is within twice or more of the maximum normal operating value, the effect is not tolerable. Moreover, the effect of sensors or actuators within the same mode behaving randomly is different than when they are scattered amongst the different modes. Even a small fraction of the nodes such as 5% behaving randomly causes significant or substantial degradation. The table y summarizes our observations with 5 nodes behaving randomly.

e) Observation 4: When sensors and actuators get stuck-at some arbitrary value, the effect is less severe than constantly varying values and degradation is tolerable.

f) Observation 5: When sensors and actuators generate random values over a wider range, the effects are far more detrimental. Even with 1 node behaving randomly, the effect is substantial. The results are summarized in table z. For the 2 nodes and 5 nodes case, the nodes were scattered amongst different modes.

C. (Intelligent) Byzantine Faults

In the following set of experiments, we program the actuators to behave in such a way that they generate the maximum possible value in a direction opposite to that of the correct control. They represent malicious nodes in the system. The nodes were chosen arbitrarily for the experiments.

g) Observation 6: Even a single byzantine node causes substantial degradation in control quality. More than one node going byzantine is intolerable.

h) Recommendation: The intolerable effects in the Boeing control scheme due to nodes generating random values and nodes exhibiting malicious byzantine behavior, can be reduced by limiting maximum actuator and sensor values. Using sensors and actuators that are sufficiently strong, warrants a redesign of control.

D. Debonding Faults

To study the effect of debonding faults on the Boeing OEP, a set of arbitrarily chosen nodes were made to perform less effectively than usual. Thus an x % debonded actuator would only apply x % of the actual force that it is supposed to apply. Likewise, a debonded sensor will only record a fraction of the actual force at its location. The debonding was specified in terms of a final effectiveness percentage and the rate of debonding.

i) Observation 7: All the nodes debonding upto 50% is tolerable. At 75% degradation is substantial. The deterioration of control quality is independent of the rate of debonding.

E. Network faults

In this section we study the effect of faults introduced by the network such as delays, on the quality of control. In the Boeing OEP, during each re-configuration cycle the nodes assigned to a group controller are changed. Also these group controllers have to report new parameters to their respective new nodes. In an ideal environment the transition from an old cycle to the new would be instantaneous. However in the presence of network delays, there arise situations in which certain group controllers have not been informed of their new assignment. Thus some nodes fall under multiple group controllers. Also there are situations in which some nodes have changed to their new assignment and parameters while some other nodes are still executing according to their earlier setting. How do nodes running in the old mode affect the nodes running in the new mode? To simulate the effect of delays, we re-implemented some of the functionalities in the Boeing OEP using threads.

j) Observation 8: Significant transient degradation is observed during reconfiguration. The effect lasts for about 0.2-0.3 seconds after re-configuration is complete.

In order to mitigate the effect of delays on the control quality, we added a synchronization service to the Boeing OEP. The goals of this service are:

1. To make sure that inspite of delays, at most one group controller is assigned to each node.
2. Nodes running in an old mode should not be affected by those in the new mode.

k) Observation 9: 10-15% of improvement is observed in the control quality after adding our synchronization service.

Thus, we see from our experiments that significant transient degradation is observed during reconfiguration. This degradation is expected to be more severe if the mode changes are frequent. Our synchronization service is helpful in mitigating this effect. The service is expected to improve the control further in the presence of synchronized clocks between nodes.

VI. Alternate Control Schemes

In the previous section we saw that nodes generating random forces and Byzantine nodes are extremely detrimental to control quality. The effects can mitigated to an extent by limiting the sensor and actuator values. But employing sufficiently strong sensors and actuators definitely warrants a different control design. As an alternative we first consider the possibility of an on-off control scheme which relies on purely local velocity feedback.

A. On-off Control Scheme With Local Velocity Feedback

[3] In this scheme the sensor at each node monitors the local velocity. Control force is binary and is applied in the opposite direction. The magnitude of this force can be chosen to vary. We chose this value to be equal to the average actuator force in the Boeing control scheme. The smaller this value, the greater the time spent in control. [3] In this scheme there is no hierarchy of control and hence there is no system-id, group controllers or configurators. Control is purely local. We studied the performance of this scheme under no faults. In order to study the effect of changing system parameters, one of our experiments was to change the plant parameters by about 20% and observe control quality using the purely local scheme. Note that in this scheme, there is no notion of a higher level configurator which monitors the plant and reports changed parameters.

l) Observation 10: The performance of the local on-off control scheme is as good as the Boeing scheme with all the 100 nodes executing the on-off local velocity fedback control law. The performance remains good even when system parameters change by +/- 20%.

It is seen that with the on-off local velocity feedback control scheme, the control is simple. The control parameter changes are avoided and communication is avoided. The effect of nodes crashing is dealt with trivially as control is purely local. This approach challenges the hierarchical control approach follwed by the Boeing OEP.

m) Observation 11: Under no faults, performance is as good as Boeing scheme with 100 nodes. However, On-Off control scheme is sensitive to sampling rate. At lower sampling frequency, force is applied in the wrong direction for a longer duration. Energy spent in control is higher.

B. Hybrid On-Off Control Scheme

Sampling is performed in the Boeing OEP nodes at 2Khz. The above mentioned on-off control scheme was found unsuitable at such low sampling rates. Hence we designed a hybrid on-off scheme in which most nodes (9 out of 10 per mode) follow the Boeing control law. The remaining nodes follow the on-off control law. The sampling rate for the on-off nodes is higher at 20Khz. We then studied the performance of this scheme with and without faults.

n) Observation 12: The energy spent in control using the hybrid on-off scheme is only marginally higher than the Boeing control scheme.

o) Observation 13: The hybrid scheme is found to be tolerant to extreme random and Byzantine faults. As shown in table zz, even in the presence of multiple byzantine nodes, the control quality is tolerable.

In these experiments at most one on-off node was chosen to be faulty. The remaining nodes were chosen randomly from different modes.

VII. Conclusions

An interesting direction for future research is to design systems that tolerate faults introduced by the middleware services due to unreliable communication channels, e.g. delays and omissions of messages. Regarding further extensions to our work, we would like to focus on fault-tolerant control theory for non linear and hybrid systems. We would also like to study the effect of continuous external perturbations on fault-tolerant control systems.

References

[1] M. Blanke, R. I. Zamanabadi, and Bogh, "Fault tolerant control systems, a holistic view," *Control Engineering Practice*, vol. 5, no. 5, 1997.

[2] "Challenge problem description for network embedded software technology (nest)," Boeing Tech Report, Boeing, Apr. 2002.

[3] Y. M. Kim, A. Arora, and V. Kulathumani, "Local distributed control of linear systems despite byzantine faults," OSU Tech Report, July 2003.